Proceedings of the Asia Pacific Industrial Engineering & Management Systems Conference 2012
V. Kachitvichyanukul, H.T. Luong, and R. Pitakaso Eds.

Solving Integrated Process Planning and Scheduling Problem with Constraint Programming

Luping Zhang, T. N. Wong†
Department of Industrial & Manufacturing Systems Engineering
The University of Hong Kong
Pokfulam Road, Hong Kong
Phone: (852) 28597055
E-mail : tnwong@hku.hk

Abstract. Process planning and scheduling are two important manufacturing functions which are usually performed sequentially. However, due to the uncertainties and disturbances frequently occurring in the manufacturing environment, the separately conducted process plan and shopfloor schedule may lose their optimality, becoming ineffective or even infeasible. Researchers have considered the potential of integrated process planning and scheduling (IPPS) to conduct the two manufacturing planning activities concurrently instead of sequentially. That is, to integrate process planning with dynamic shopfloor scheduling to cope with the realtimeshopfloor status.The IPPS problem is very complex and it has been regarded as an NP-hard problem. Many researchers have attempted to solve the IPPS problem with intelligent approaches such as meta-heuristics and agent-based negotiation. In this paper, a constraint programming-based approach is proposed and implemented in the IPPS problem domain. Constraint programming (CP) features great modeling capabilities to reflect complex constraints of a problem, and there is a great potential for CP to be used to solve IPPS problems. The approach is implemented and tested on the IBM ILOG platform, and experimental results show that the CP can handle the IPPS problem efficiently and effectively.

Keywords: Constraint programming; process planning; job-shop scheduling; IPPS

1. INTRODUCTION

Process planning and scheduling are two crucial functions in the manufacturing environment. They are closely related to each other but usually performed sequentially, which sometime renders the process plan and scheduling infeasible. Furthermore, the sequential and separate conduction of process planning and scheduling cannot satisfy the dynamic manufacturing environment with uncertainties and disturbances. That is why the integrated process planning and scheduling (IPPS) is proposed to combine the process planning and scheduling together into consideration, responding to the resource restriction and dynamic manufacturing situations.

To improve the optimality of the process plans and the schedule, some researchers have attempted to solve the IPPS problem with exact optimization methods and advanced search techniques.In recent years, meta-heuristics are widely considered as promising approaches to tackle the IPPS problems. A similar problem, job-shop scheduling problem has been addressed using simulated annealing(Steinhofel, Albrecht, & Wong, 1999). The IPPS problems of a simple version, without consideration of alternative processes have also been tackled by Morad and Zalzala with Genetic Algorithms(Morad & Zalzala, 1999).In 2007, Li & McMahonalso used the simulated annealing and combined the process planning and scheduling together(Li & McMahon, 2007).A particle swarm optimisation approach was adopted to solve the IPPS problems later(Guo, Li, Mileham, & Owen, 2009).Besides meta-heuristics, agent-based approaches have also been proposed for the IPPS (Wong, Leung, Mak, & Fung, 2006a; Wong, Leungy, Mak, & Fung, 2006). Meta-heuristics and agent-based approaches have been found to be more effective and efficient for larger IPPS problems, with the objective to find near-optimal solutions in a reasonable time.

† :Corresponding Author

The application of constraint programming (CP) to scheduling problems has been systematically demonstrated by Baptiste and et al. in their book (Baptiste, Le Pape, & Nuijten, 2001).The CP has also been implemented by other researchers to solve the multi-machine assignment scheduling problem (Sadykov & Wolsey, 2006), and job-shop scheduling problems (Beck, Feng, & Watson, 2011).Although some researchers have implemented the CP in both planning and scheduling problems(Timpe, 2002), we have not yet come across any publications on the successful application of CP in IPPS.

In this paper, we pursue the exploration of CP to tackle the IPPS problems. A CP-based IPPS model is proposed. And IBM ILOG is adopted to program and solve our IPPS model. A graph-based representation is given in section 2to illustrate the IPPS problem clearly, and our CP-based model is closely related to the graphic representation. After that, the CP algorithm for our IPPS problem is depicted in detail in section 3.And then, the proposed approach is tested on IBM ILOG with benchmark problems, and experiments results are also given in section 4. Final conclusions are demonstrated in section 5.

2. GRAPH-BASED REPRESENTATION OF IPPS PROBLEM

Usually, the IPPS problem features **n** jobs to be processed on **m** machines, each job consistsof a group of operations which may involve alternate processes and alternate machines, and precedencerelationships may exist between different operations(Wong, Leung, Mak, & Fung, 2006b).Process planning alternatives of the IPPS problem can be represented graphically with the AND/OR graph. For example, the AND/OR graphs in Figure 1 show the respective process planning information for 2 jobs with altogether 11 operations, where each node denotes an operation. An or-relationship is declared for job 2.

The following notations are used to represent primitive elements of the IPPS problem.

J_j : a job with index number j;

m_k : a machine with the index number k;

$O_{j,i}$: an operation of job J_j with index number i;

$T(O_{j,i})$: a given type of the operation $O_{j,i}$;

$O_{j,i}^k$: operation $O_{j,i}$ to be processed on machine m_k;

$\tau(O_{j,i}^k)$: theprocessing time of $O_{j,i}^k$;

OR_r : an or-relationship with index number r. It can be expressed by attaching three ordered operations $OR_r(O_{j_0,i_0}, O_{j_1,i_1}, O_{j_2,i_2})$, which indicates an or-relationship exists among operations $O_{j_0,i_0}, O_{j_1,i_1}, O_{j_2,i_2}$;

Seq_i : a sequence relationship with index number i. It can be expressed by attaching a pair of operations $Seq_i(O_{j_1,i_1}, O_{j_2,i_2})$, which means O_{j_1,i_1} is an immediate predecessor of O_{j_2,i_2};

$C_{\max}$: themakespan of the IPPS instance;

J : the set of jobs of the IPPS problem $\{J_j\}$;

O : the set of operations of the IPPS problem $\{O_{j,i}\}$;

P : the set of operations with processing machine allocated $\{O_{j,i}^k\}$;

M : the set of available machines $\{m_k \mid k \neq 0\}$, and m_0 is set to be a special dummy machine;

R : the set of all or-relationships $\{OR_r(O_{j_0,i_0}, O_{j_1,i_1}, O_{j_2,i_2})\}$;

SEQ : the set of all sequence relations defined in the IPP problem $\{Seq_i(O_{j_1,i_1}, O_{j_2,i_2})\}$.

Regarding the IPPS problem discussed in this paper, each machine can process one operation at a time, and on operation cannot be interrupted when processing begins. Hence, the scheduling we discussed here is a typical disjunctive non-preemptive scheduling problem. As a result, a disjunctive graph model ***D=(O,A,R)*** can be used to aid the representation of the IPPS problem, as shown in Figure 2. **O** is a set of nodes which represent operations, and a set of candidate machines for processing each operation $O_{j,i}$ is allocated. **A** is a set of directed arcs, depicting the sequence relationships between different operations. If a directed arc starts from O_{j_1,i_1} and points to O_{j_2,i_2}, it means O_{j_1,i_1} is an immediate predecessorof O_{j_2,i_2}.Using notations mentioned above, the sequence relationship can be expressed by $Seq_i(O_{j_1,i_1}, O_{j_2,i_2})$. **R** is a set of dummy nodes, indicating the or-relationship of alternative processes. Figure 2 presents the disjunctive graph for the 2 jobs given in Figure 1. As shown in this diagram, node OR_1 indicates that the process trail $(O_{2,8}, O_{2,9})$ and $(O_{2,10})$ are optional and only one trail can be chosen in a solution. And the or-relationship can be writtenby $OR_1(O_{2,7}, O_{2,8}, O_{2,10})$, where $O_{2,7}$ is the immediate predecessor of node OR_1, and $O_{2,8}$ and $O_{2,10}$ are two immediate successors of this or-node. S_j and F_j are two other kinds of dummy nodes to indicate the starting point and ending point of job J_j, within which S_0 is the starting point of the whole IPPS instance. For thepurpose of consistency, the starting node S_j and terminating node F_j are regarded to be special operations. To distinguish S_j and F_j from other nodes, index numbers of S_j and F_j are set to be 0 and 9999 (or a very large integer) respectively, that is:

$$S_j = O_{j,0}, \quad F_j = O_{j,9999} \quad (1)$$

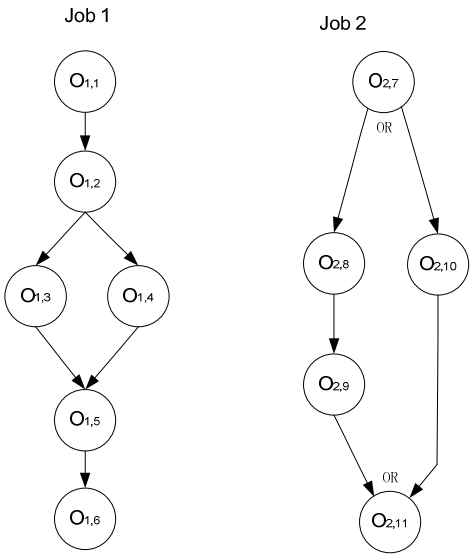

Figure 1: AND/OR graph of an sample IPPS instance

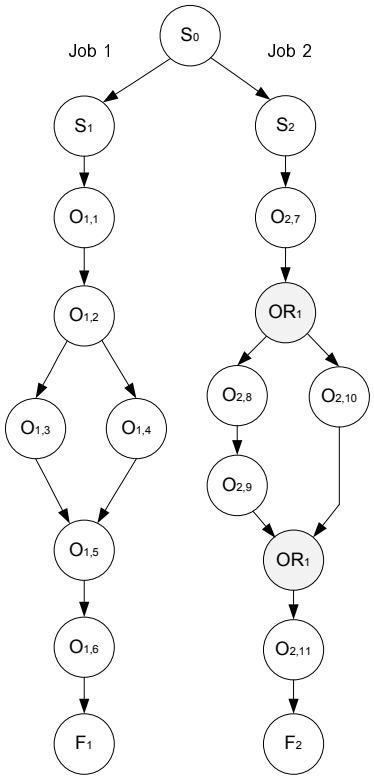

Figure 2: Disjunctive graph for the IPPS problem.

And a dummy machine m_0 is allocated to process the dummy nodes, with processing time set to be 0:

$$\tau(O^0_{j,0}) = \tau(O^0_{j,9999}) = 0 \quad (2)$$

3.CP ALGORITHM FOR THE IPPS PROBLEM

The CP algorithm consists of several modules. Based on the IBM ILOG open programming language (OPL), three basic modules are essential to establish a CP model: variables, objective function, and the definition of constraints.

3.1 Variables

To begin with, a primitive type of variablesinterval should be defined. In this paper, the interval represents the interval of time during which an operation is processed. An interval can be characterized by its starting time, its ending time, and its size. In this paper, since the IPPS is defined as a non-preemptive scheduling problem, the size of an interval equals to the difference between its starting time and ending time. However, before a solution is obtained, the starting time and ending time of an operation's interval remains unknown, which means the allocation of an operation's interval on a schedule is not determined.

Furthermore, an interval should be assigned to each operation when resolving the problem. However, since alternative machines and alternative processes exist, it is not required to map all intervals onto the schedule. As a result, another important feature of an interval is that it can be optional.

Intervals are needed to be assigned to each nodes appearing in the disjunctive graph. The overall definition of variables issummarized as follows:

Interval *st* size 0;

Interval $p[O^k_{j,i}]$ optional size $\tau(O^k_{j,i})$ $\forall O^k_{j,i} \in P$;

Interval $ops[O_{j,i}]$ optional $\forall O_{j,i} \in O$;

Interval $or[OR_r]$ optional $\forall R_r \in R$;

Sequence $mch[m_{k_0}] = \{p[O^k_{j,i}] \mid k = k_0\}$ $\forall m_{k_0} \in M$;

Sequence $job[J_{j_0}] = \{p[O^k_{j,i}] \mid j = j_0\}$ $\forall J_{j_0} \in J$.

st is a dummy interval assigned to the start node of the whole IPPS problem, which is S_0 in Figure 2.

In the above definition, the variable $f[E]$ represents a collection of variables $f[]$, containing a data member E as its implementation object.

$p[O^k_{j,i}]$ reflects the interval needed to process operation $O_{j,i}$ on machine m_k, so the size is equal to the processing time of $O^k_{j,i}$, which is $\tau(O^k_{j,i})$.

$ops[O_{j,i}]$ is the interval for the operation $O_{j,i}$ without allocation of processing machine. When an $p[O^k_{j,i}]$ is present, the $ops[O_{j,i}]$ is present with the same starting point and ending point as those of $p[O^k_{j,i}]$.

$or[OR_r]$ is a dummy interval. It only guarantees that when its predecessor operation O_{j_0,i_0} is present, one of its successor operations O_{j_1,i_1} and O_{j_2,i_2} will be present as well.

Sequence is another primitive variable type to indicate a set of ordered interval variables. $mch[m_{k_0}]$ is a subset of $p[O^k_{j,i}]$ including those intervals whose operations are processed on m_k. $job[J_{j_0}]$ is another subset of $p[O^k_{j,i}]$ containing intervals whose operations belong to job J_{j_0}.

3.2Objective Function

Makespan is a crucial criterion for the planning and scheduling problem. Here minimizing the makespan is regarded to be the only objective for the CP algorithm. According to the IPPS model described in section 2, each job is terminated with a dummy node F_j.The makespan of the IPPS problem can be easily obtained by the following function:

$$C_m = \max\{startof(ops[O_{j,9999}]) : \forall J_j \in J\} \quad (3)$$

where $startof(I)$ is an expression to access the start point of interval I.

As a result, in one experiment, the objective is to minimize the makespan, which is:

$$\min C_m = \min \max\{startof(ops[O_{j,9999}]) : \forall J_j \in J\} \quad (4)$$

3.3The Definition of Constraints

Variables given above must subject to all constraints listed as follows:

$$endbeforestart(st, ops[O_{j,0}]) : \forall J_j \in J \quad (5)$$

$$endbeforestart(ops[firstof(Seq_i)], ops[nextof(Seq_i)]) : \forall Seq_i \in SEQ \quad (6)$$

$$noOverlap(mch[m_k]) : \forall m_k \in M \quad (7)$$

$$noOverlap(job[J_j]) : \forall J_j \in J \quad (8)$$

$$alternative(ops[O_{j_0,i_0}],\{prs[O^k_{j,i}] \mid j=j_0, i=i_0\}) : \forall O_{j_0,i_0} \in O \quad (9)$$

$$alternative(or[OR_r],\{ops[nextof(OR_r)], ops[thirdof(OR_r)]\}) : \forall OR_r \in R \quad (10)$$

Function (5) restricts that the algorithm must start searching from the overall start point S_0 of the IPPS problem, where $endbeforestart(I_1, I_2)$ is a constraint function to indicate that the end of interval I_1 is less than or equal to the start of interval I_2. Or it can be expressed by the relationship $endof(I_1) \le startof(I_2)$, where $startof(I)$ and $endof(I)$ are two expressions to access the start and end of the interval I respectively.

In function (6), $firstof(Seq_i)$ and $nextof(Seq_i)$ are two expressions to access the first and second elements of the operation pairs stated in Seq_i. For example, if Seq_i represents the sequence relationship $Seq_i(O_{j_1,i_1}, O_{j_2,i_2})$, the $firstof(Seq_i)$ and $nextof(Seq_i)$ will return the operation O_{j_1,i_1} and O_{j_2,i_2} respectively. Hence, the function (6) realizes the sequence constraints between different operations.

In functions (7) and (8), $nooverlap(s)$ is a constraint to prevent all intervals included in sequence s from overlapping. Hence function (7) restricts that a machine can only process one operation at a time, and function (8) restricts that different operations of the same job are not permitted to be processed simultaneously.

In functions (9) and (10), $alternative(I, \{I_1, I_2 .... I_n\})$ models an exclusive alternative interval among $\{I_1, I_2 .... I_n\}$.If interval I is present, exactly one interval among $\{I_1, I_2 .... I_n\}$ should be present, with the start point and end point equal to those of I respectively. And in function (10), expressions $nextof(OR_r)$ and $thirdof(OR_r)$ can return the second and third operations stated in the operation group $OR_r(O_{j_0,i_0}, O_{j_1,i_1}, O_{j_2,i_2})$, which are O_{j_1,i_1} and O_{j_2,i_2} respectively. So function (9) restricts that one operation can only be processed once on a specific machine.

Function (10) restricts that only one of the alternative processes can be chosen.

3.4Constraint Propagation

To reduce the search space and accelerate the searching process, a constraints propagation method is designed in our approach.

The operations in an IPPS problem are classified into two categories: compulsory and optional operations. Compulsory operations refer to those which have to be processed in order to accomplish a job. For example, all operations other than $O_{2,8}$, $O_{2,9}$ and $O_{2,10}$ shown in Figure 2 are compulsory operations. Optional operations refer to those operations without which the job can also be accomplished. Usually, operations on alternative process trails are optional. For example, $O_{2,8}$, $O_{2,9}$ and $O_{2,10}$ in Figure 2 are optional operations.

Here we use $T(O_{j,i})$ to indicate the type of each operation. It is defined that:

$$T(O_{j,i}) = \begin{cases} 0 & when\ O_{j,i}\ is\ compulsory \\ 1 & when\ O_{j,i}\ is\ optional \end{cases} \quad (11)$$

Obviously, if an interval $ops(O_{j,i})$ is not present in a solution when $T(O_{j,i}) = 0$, the solution can be eliminated from the search space. Actually, the search space can be substantially shrunk due to the upper constraints.

Another situation occurs in the group of optional operations. Take the optional operations in Figure 2 as an example. Although $O_{2,8}$ and $O_{2,9}$ are both optional operations, $O_{2,9}$ has to be present if $O_{2,8}$ is present in the final solution. So we can propagate a new constraint to associate $O_{2,8}$ with $O_{2,9}$. Here we designed a new data element to record this associative relationship, which can be denotedas $Asso_a$, where a is the index number of the associative relationship. And similar to Seq_i, a pair of operations can be attached to $Asso_a$ to indicate the specific associative operations. So we can write $Asso_a(O_{j_1,i_1}, O_{j_2,i_2})$, which means if O_{j_1,i_1} is present, O_{j_2,i_2} have to be present as well. In addition, a set $ASSO$ can be defined to restore all of such kind of relations, which means $ASSO = \{Asso_a(O_{j_1,i_1}, O_{j_2,i_2})\}$.

Moreover, another situation may exist: what happens if $O_{2,7}$ in Figure 2 is also on an alternative process trail? That means $O_{2,7}$ turns to be optional in that circumstance. So another associative relation may exist between the or-node OR_1 and $O_{2,7}$: when $O_{2,7}$ is present, OR_1 has to be present as well, and vice versa. As a result, another constraint needs to be defined between the or-node and its immediate predecessor. As defined in section 2, OR_r can be represented by $OR_r(O_{j_0,i_0}, O_{j_1,i_1}, O_{j_2,i_2})$, in which O_{j_0,i_0} is the immediate predecessor of OR_r and it can be accessed from the expression $firstof(OR_r)$.

As discussed above, the following constraints can be propagated:

$$presenceOf(st) => presenceOf(O_{j,i}) : \forall T(O_{j,i}) = 0 \quad (12)$$

$$presenceOf(ops[firstof(Asso_a)]) => presenceOf(ops[nextof(Asso_a)]) : \forall Asso_a \in ASSO \quad (13)$$

$$presenceOf(ops[firstof(OR_r)]) => presenceOf(or[OR_r]) : \forall OR_r \in R \quad (14)$$

where $presenceOf(I_1) => presenceOf(I_2)$ means that the presence of interval I_1 will enforce the presence of interval I_2 .Function (12) restricts that all compulsory operations have to be present. Function (13) indicates that those operations with an associative relationship should be present or absent together. And function (14) associates the or-node with its immediate predecessor.

4. EXPERIMENTS AND DISCUSSIONS

The IBM ILOG CPLEX Optimization Studio 12.4 edition is introduced to program and handle the CP-based IPPS model. Experiments are conducted on a Dell desktopPC with i7-2600 CPU and 6M RAM. For a benchmark purpose, the test bed with 18 jobs, up to 300 operations, and 24 test problems designed by Kim, et al. is adopted for experiments(Kim, Park, & Ko, 2003).

FailLimit is a crucial parameter for the control of search termination in IBM ILOG, which defines a limit restricting the number of failures during the search. Here the FailLimit is set to be 200,000 for all problems tested in our experiments. Each problem of the test bed is executed with 6 runs and the average of the 6 simulation runs is taken. After completion of the experimental runs, the average results generated by CP-based approach are compared with those generated by two evolution-base algorithms, which are the cooperative co-evolutionary genetic algorithm (CCGA)(Potter & De Jong, 1994), and the symbiotic evolutionary algorithm (SEA)(Kim et al., 2003).

Table 1 accounts the mean makespan of 10 problems generated by CCGA, SEA and CP-based approach. The CPU time needed for the CP-based approach is also given in the table for reference.

Problems 1 to 6 are relatively simple IPPS instances with 6 jobs included. Problems 21 to 24 arecomplex IPPS instances with more than 15 jobs and near 300 operations included. Firstly, Table 1 demonstrates that CP-based approach has better performance on the IPPS, since its resultsare better than those of theother two algorithms in 9 out of 10 problems of the test bed. Secondly, it can be found that the CP-based approach performs much better than the other two algorithms when solving larger-size problems. The improved rate exceeds 20% for the problem 23 and 24, which means the proposed approachcan be implemented to resolve complex combinatorial optimization problems. At last, table 1 reveals that the CP-based approach is pretty efficient since the time consumed for computation does not grow rapidly when the problem domain expands quickly.

The gantt chart shown in Figure 3 depicts a schedule for the problem 24 with 18 jobs and 300 operations, which reveals a very high machine utilization. The accuracy and feasibility of the proposed approach can also be verified.

Figure 4(a) and Figure 4(b) illustrate the convergence curves of the respective makespans of problems 2 and 24.The x-axis and y-axis represent the time and makespan's value respectively. Actually, the convergence curves for problems 3 to 6 feature the same trend as shown in Figure 4(a), which shows that a very good solution can be found within 10 seconds. Figure 4(b) shows another convergence trend for a complex IPPS instance. It reveals that a good solution can also be found with around 10 seconds. However, it also shows that the CP-based approach possesses potentials to explore the search space and continuously find better solutions if the FailLimit is big enough.

The propagated constraints may provide substantial contribution to such results, since invalid solutions can be quickly verified and eliminated from the search space, and the size of search space can be dropped to a great extent at the early stage of the searching procedure.

5. CONCLUSIONS

In this paper, constraint programming (CP) is implemented to solve the IPPS problem. A complex IPPS model with alternative machines and processes is established based on the CP language. Variables and constraints are clearly defined. Furthermore, a method of constraint propagation is given in order to enhance the efficiency of the proposed approach. Experiments on a set of benchmark test bed problems are conducted, which reveals the feasibility and excellent performance of the CP in the domain of IPPS problems.

However, another phenomenon is also revealed in our experiments. Results of different experiments on the same problem feature a high level of similarity. Final objective values for the same problem turn out to be the same in repeated experiments. Moreover, the convergence trend curves for the same problem in repeated experiments follow a fairly similar trail. So it might be assumed that the search procedure of the IBM ILOG CPLEX Optimization Studio might be non-randomized. It might prohibit the further improvement of the CP's performance in the field of IPPS problems. In this regard, the combination of CP

and heuristic methods to solve the IPPS problem might be a good research direction.

Table 1: The comparison of averagemakespan

Problem	SEA	CCGA	CP-based	Best	Improved rate (%)	CPU time (s)
1	437.6	470.4	427	CP-based	2.4	6
2	349.7	369.2	343	CP-based	1.9	29
3	355.2	382	345	CP-based	2.9	29
4	306.2	321.5	306	CP-based	0.1	29
5	323.7	337.8	326	SEA	-0.7	29
6	443.8	485.6	427	CP-based	3.8	9
21	483.2	534	427	CP-based	11.6	19
22	548.3	587.5	454	CP-based	17.2	33
23	507.5	557.9	397	CP-based	21.8	34
24	602.2	633.3	478	CP-based	20.6	36

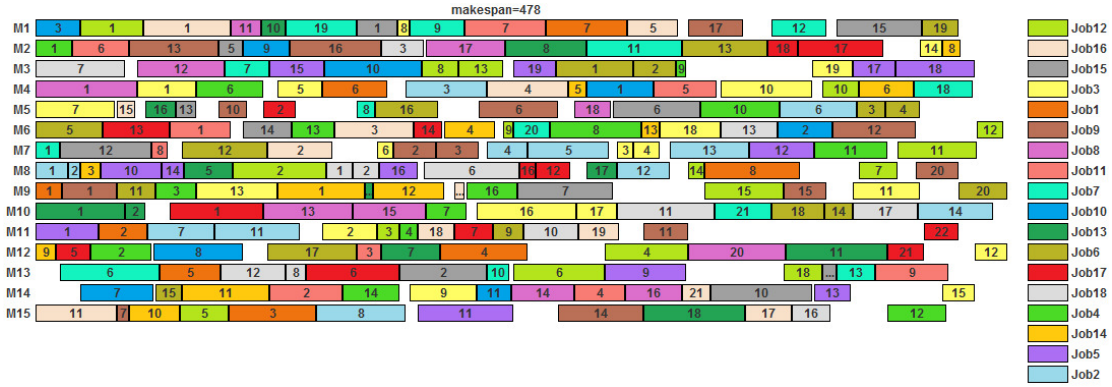

Figure 3: A gantt chart for problem 24

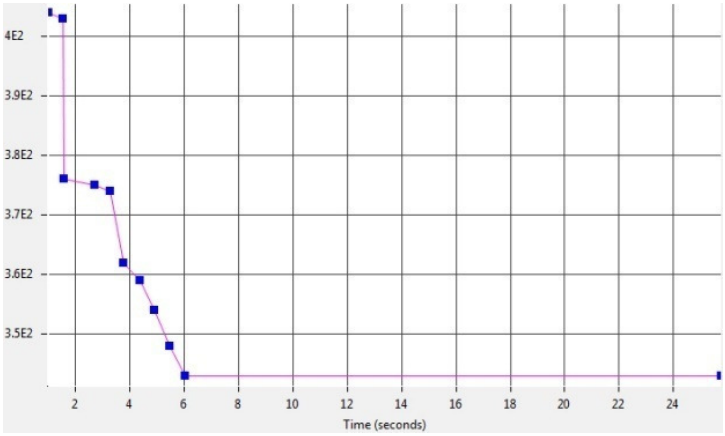

Figure 4(a): Makespan's convergence curve of problem 2

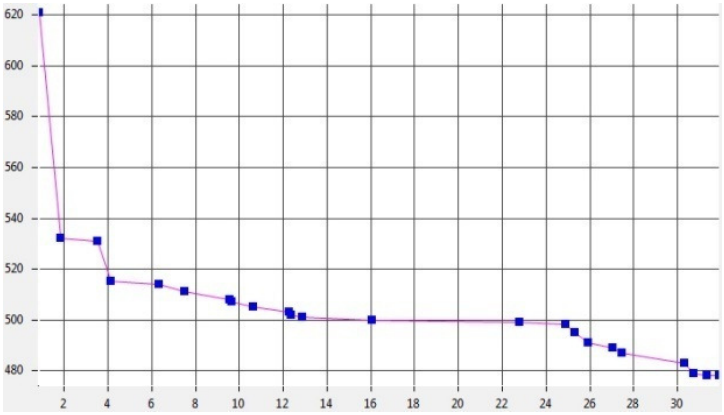

Figure 4(b): Makespan's convergence curve of problem 24

ACKNOWLEDGMENT

The work described in this paper is fully supported by a grant from the Research Grants Council of Hong Kong (Project Code HKU 718809E).

REFERENCES

Baptiste, P., Le Pape, C., & Nuijten, W. (2001). *Constraint-based scheduling : applying constraint programming to scheduling problems*. Boston: Kluwer Academic.

Beck, J. C., Feng, T. K., & Watson, J. P. (2011). Combining Constraint Programming and Local Search for Job-Shop Scheduling. *Informs Journal on Computing, 23*(1), 1-14.

Guo, Y. W., Li, W. D., Mileham, A. R., & Owen, G. W. (2009). Optimisation of integrated process planning and scheduling using a particle swarm optimisation approach. *International Journal of Production Research, 47*(14), 3775-3796.

Kim, Y. K., Park, K., & Ko, J. (2003). A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling. *Computers & Operations Research, 30*(8), 1151-1171.

Li, W. D., & McMahon, C. A. (2007). A simulated annealing-based optimization approach for integrated process planning and scheduling. *International Journal of Computer Integrated Manufacturing, 20*(1), 80-95.

Morad, N., & Zalzala, A. (1999). Genetic algorithms in integrated process planning and scheduling. *Journal of Intelligent Manufacturing, 10*(2), 169-179.

Potter, M. A., & De Jong, K. A. (1994). A cooperative coevolutiouary approach to function optimization. *Parallel Problem Solving from Nature - Ppsn Iii - International Conference on Evolutionary Computation, Proceedings, 866*, 249-257.

Sadykov, R., & Wolsey, L. A. (2006). Integer programming and constraint programming in solving a multimachine assignment scheduling problem with deadlines and release dates. *Informs Journal on Computing, 18*(2), 209-217.

Steinhofel, K., Albrecht, A., & Wong, C. K. (1999). Two simulated annealing-based heuristics for the job shop scheduling problem. *European Journal of Operational Research, 118*(3), 524-548.

Timpe, C. (2002). Solving planning and scheduling problems with combined integer and constraint programming. *Or Spectrum, 24*(4), 431-448.

Wong, T. N., Leung, C. W., Mak, K. L., & Fung, R. Y. K. (2006a). An agent-based negotiation approach to integrate process planning and scheduling. *International Journal of Production Research, 44*(7), 1331-1351.

Wong, T. N., Leung, C. W., Mak, K. L., & Fung, R. Y. K. (2006b). Dynamic shopfloor scheduling in multi-agent manufacturing systems. *Expert Systems with Applications, 31*(3), 486-494.

Wong, T. N., Leungy, C. W., Mak, K. L., & Fung, R. Y. K. (2006). Integrated process planning and scheduling/rescheduling - an agent-based approach. *International Journal of Production Research, 44*(18-19), 3627-3655.